STUDENT ID NO

MULTIMEDIA UNIVERSITY

MODULE TEST 1 & 2

TRIMESTER 1, 2020/2021

ECE3166 ADVANCED MICROPROCESSORS
(All Majors)

25 SEPTEMBER 2020
11:00A.M. – 12:00P.M.
(1 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of SIX pages with TWO Questions only.

2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.

3. Please print all your answers in your own answer papers.

4. Scan and submit your answer papers to Google Classroom.

Question 1

(a) Intel 8086 microprocessor is the first member of the x86 family of microprocessors.

i. State the size of data bus and address bus of the 8086 microprocessor in bits. [2 marks]

ii.. Determine the maximum size of memory address space in the 8086 microprocessor in bytes and list the lowest and highest addresses in the memory space in **hexadecimal**. [4 marks]

(b) The decimal data 24681357_{10} is stored in the memory of an Intel 8086 microprocessor starting from address 13680H. Illustrate how this data is stored in the memory by labelling the memory addresses and their corresponding contents in **hexadecimal**. [4 marks]

(c) Determine the memory capacity of each microprocessor in Table Q1(c) in **bytes**. Rank the microprocessors in descending order (big to small) according to their memory capacity. [5 marks]

Table Q1(c)

Microprocessor	Width of memory block (bits)	Size of address bus (bits)
P	8	20
Q	32	16
R	16	12
S	4	18

(d) A 16-bit 2s complement number is stored in the memory of an Intel 8086 microprocessor as shown in Table Q1(d). Convert this number into its decimal equivalent. Is this an aligned or misaligned word of data? [4 marks]

Table Q1(d)

Address	Content
12502H	10111010_2
12503H	10101011_2

(e) Intel 80386 is the first IA-32 microprocessor. Determine the number of active segments and the maximum size of active memory when the 80386DX processor is operating in real-mode. Determine the starting and ending addresses of the code segment and data segment if the contents of the segment registers are as shown Table Q1(e). Are the code segment and data segment contiguous, overlapping or disjoint? [6 marks]

Table Q1(e)

Register	Contents
CS	1200H
DS	2100H

Continued …

Question 2

(a) If the initial contents of registers are as given in Table Q2(a), determine the addressing mode and physical address of the DESTINATION operand in each of the following instructions:

i. MOV ES:[1024H], AX [4 marks]

ii. MOV [BX][DI]+357H, CH [4 marks]

Table Q2(a): Contents of registers

Register	Contents
AX	0101H
BX	2903H
CX	3561H
DI	5AA5H

Register	Contents
CS	1000H
DS	2000H
ES	3000H
SS	4000H

Note: All workings must be shown.

(b) An x86 assembly instruction sequence is given in Table Q2(b). Complete and reproduce this table in your answer paper. State the contents of the destination registers in **hexadecimal**. [6 marks]

Table Q2(b): An instruction sequence

Instruction	Contents of destination register
MOV AX, 1357H	AX = 1357H
MOV BX, 0A55AH	BX = A55AH
XOR AL, 6BH	
NOT BH	
SAR BL, 1	
ROL AH, 1	

(c) Write a real mode x86 instruction sequence to perform the following arithmetic operation and store the final result in register BX.

$$9368_{10} + 39_{10} \times 28_{10}$$ [6 marks]

(d) A real mode x86 instruction sequence is given below:

```
MOV  SI, 1350H
MOV  DI, 2460H
MOV  CX, 3
CLD
REP  MOVSW
```

Initial contents of the segment registers are given as:

CS = 1200H, DS = 230H, ES = 3400H. SS = 4500H

i. How many bytes of data will be moved in this instruction sequence? [2 marks]

ii. Analyze the instruction sequence and determine the physical addresses of the affected memory locations. [3 marks]

End of Paper

Appendix A

Instruction Set Summary for Intel 8086/80186/80286/80386/80486

AAA - Ascii Adjust for Addition
AAS - Ascii Adjust for Subtraction
ADC - Add With Carry
ADD - Arithmetic Addition
AND - Logical And
BSF - Bit Scan Forward (386+)
BSR - Bit Scan Reverse (386+)
BSWAP - Byte Swap (486+)
BT - Bit Test (386+)
BTC - Bit Test with Compliment (386+)
BTR - Bit Test with Reset (386+)
BTS - Bit Test and Set (386+)
CALL - Procedure Call
CBW - Convert Byte to Word
CDQ - Convert Double to Quad (386+)
CLC - Clear Carry
CLD - Clear Direction Flag
CLI - Clear Interrupt Flag (disable)
CLTS - Clear Task Switched Flag (286+ privileged)
CMC - Complement Carry Flag
CMP - Compare
CMPS - Compare String (Byte, Word or Doubleword)
CMPXCHG - Compare and Exchange
CWD - Convert Word to Doubleword
CWDE - Convert Word to Extended Doubleword (386+)
DAA - Decimal Adjust for Addition
DAS - Decimal Adjust for Subtraction
DEC - Decrement
DIV - Divide
ENTER - Make Stack Frame (80188+)
ESC - Escape
HLT - Halt CPU
IDIV - Signed Integer Division
IMUL - Signed Multiply
IN - Input Byte or Word From Port
INC - Increment
INT - Interrupt
INTO - Interrupt on Overflow
INVD - Invalidate Cache (486+)
INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)
IRET/IRETD - Interrupt Return
Jxx - Jump Instructions Table
JCXZ/JECXZ - Jump if Register (E)CX is Zero
JMP - Unconditional Jump
LAHF - Load Register AH From Flags
LAR - Load Access Rights (286+ protected)
LDS - Load Pointer Using DS
LEA - Load Effective Address
LES - Load Pointer Using ES
LFS - Load Pointer Using FS (386+)
LGDT - Load Global Descriptor Table (286+ privileged)
LIDT - Load Interrupt Descriptor Table (286+ privileged)
LGS - Load Pointer Using GS (386+)
LLDT - Load Local Descriptor Table (286+ privileged)
LMSW - Load Machine Status Word (286+ privileged)
LOCK - Lock Bus
LODS - Load String (Byte, Word or Double)

LOOP - Decrement CX and Loop if CX Not Zero
LOOPE/LOOPZ - Loop While Equal / Loop While Zero
LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
LSL - Load Segment Limit (286+ protected)
LSS - Load Pointer Using SS (386+)
LTR - Load Task Register (286+ privileged)
MOV - Move Byte or Word
MOVS - Move String (Byte or Word)
MOVSX - Move with Sign Extend (386+)
MOVZX - Move with Zero Extend (386+)
MUL - Unsigned Multiply
NEG - Two's Complement Negation
NOP - No Operation (90h)
NOT - Logical inversion
OR - Inclusive Logical OR
OUT - Output Data to Port
POP - Pop Word off Stack
POPA/POPAD - Pop All Registers onto Stack (80188+)
POPF/POPFD - Pop Flags off Stack
PUSH - Push Word onto Stack
PUSHA/PUSHAD - Push All Registers onto Stack (80188+)
PUSHF/PUSHFD - Push Flags onto Stack
RCL - Rotate Through Carry Left
RCR - Rotate Through Carry Right
REP - Repeat String Operation
REPE/REPZ - Repeat Equal / Repeat Zero
REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
RET/RETF - Return From Procedure
ROL - Rotate Left
ROR - Rotate Right
SAHF - Store AH Register into FLAGS
SAL/SHL - Shift Arithmetic Left / Shift Logical Left
SAR - Shift Arithmetic Right
SBB - Subtract with Borrow/Carry
SCAS - Scan String (Byte, Word or Doubleword)
SETS - Set if Signed (386+)
SETNS - Set if Not Signed (386+)
SETC - Set if Carry (386+)
SETNC - Set if Not Carry (386+)
SETO - Set if Overflow (386+)
SETP/SETPE - Set if Parity / Set if Parity Even (386+)
SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)
SGDT - Store Global Descriptor Table (286+ privileged)
SIDT - Store Interrupt Descriptor Table (286+ privileged)
SHL - Shift Logical Left
SHR - Shift Logical Right
SHLD/SHRD - Double Precision Shift (386+)
SLDT - Store Local Descriptor Table (286+ privileged)
SMSW - Store Machine Status Word (286+ privileged)
STC - Set Carry
STD - Set Direction Flag
STI - Set Interrupt Flag (Enable Interrupts)
STOS - Store String (Byte, Word or Doubleword)
STR - Store Task Register (286+ privileged)
SUB - Subtract
TEST - Test for Bit Pattern
WBINVD - Write-Back and Invalidate Cache (486+)
XCHG - Exchange
XOR - Exclusive OR

Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVSB/MOVSW	((ES)0 + (DI)) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare string	CMPSB/CMPSW	Set flags as per ((DS)0 + (SI)) − ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCASB/SCASW	Set flags as per (AL or AX) − ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODSB/LODSW	(AL or AX) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store string	STOSB/STOSW	((ES)0 + (DI)) ← (AL or AX) (DI) ← (DI) ± 1 or 2	None

Prefix	Used with:	Meaning
REP	MOVS STOS	Repeat while not end of string $CX \neq 0$
REPE/REPZ	CMPS SCAS	Repeat while not end of string and strings are equal $CX \neq 0$ and $ZF = 1$
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal $CX \neq 0$ and $ZF = 0$